Solving the Constrained Problem in Model Predictive Control Based Motion Cueing Algorithm with a Neural Network Approach

C. Rengifo[1,2], J.-R. Chardonnet[2], D. Paillot[3], H. Mohellebi[1] and A. Kemeny[1,2]

(1) Renault, Virtual Reality and Immersive Simulation Center, 78288 Guyancourt, France, e-mail: {carolina.c.rengifo, hakim.mohellebi, andras.kemeny}@renault.com
(2) LISPEN EA7515, Arts et Métiers, HESAM, Université Bourgogne Franche-Comté, Institut Image, 71100 Chalon-sur-Saône, France, e-mail: carolina.c.rengifo@renault.com, jean-remy.chardonnet@ensam.eu
(3) Université de Bourgogne, LISPEN EA7515, Arts et Métiers, HESAM, Université Bourgogne Franche-Comté, Institut Image, 71100 Chalon-sur-Saône, France, e-mail: damien.paillot@u-bourgogne.fr

Abstract - *Because of the critical timing requirement, one major issue regarding model predictive control-based motion cueing algorithms is the calculation of real-time optimal solutions. In this paper, a continuous-time recurrent neural network-based gradient method is applied to compute the optimal control action in real time for an MPC-based MCA. We demonstrate that by implementing a saturation function for the constraints in the decision variables and a regulation for the energy function in the network, a constrained optimization problem can be solved without using any penalty function. Simulation results are included to compare the proposed approach and substantiate the applicability of recurrent neural networks as a quadratic programming solver. A comparison with another QP solver shows that our method can find an optimal solution much faster and with the same precision.*

Keywords: *Optimization, recurrent neural networks, MPC motion cueing algorithm, real-time simulator, penalty method.*

Introduction

Driving simulators are necessary tools to evaluate human behaviors in driving situations, to validate benefits of advanced systems and autonomous driving and to simulate driving scenarios in safe and controlled environments. Motion commands such as accelerations and rotations cannot be sent directly from the model of the simulated vehicle to the platform, because of specific physical constraints and limits in the workspace.

For this reason, different control strategies called Motion Cueing Algorithms (MCA) are implemented, which aim to better reproduce all the signals coming from the simulated vehicle model while keeping the platform within its boundaries. There are mainly four motion cueing strategies, three of them being filter-based control strategies: classical, adaptative, optimal, and one based on prediction [Gar10]. Past works have shown that model predictive control (MPC)-based MCA [Dag04] is the most suitable strategy, as it uses optimization to track the reference, respects the constraints and maximizes the workspace. Moreover the MPC procedure provides easier tuning and is more robust [Neh06, Fan12, Beg12, Ven16].

In MPC-based MCA the system dynamics varies quickly and the numerical solution that is required in real time is often seen as an obstacle for normal solvers. This optimization problem solution in real time has been studied intensively [Goo06]. One promising approach is to use artificial recurrent neural networks (RNN) as a method to compute dynamic optimization based on hardware implementation. The recurrent neural network capabilities such as high parallelism, adaptability, and hardware-based circuit implementation are recognized by researchers and are used in many different application areas including: control system design, associative memory, mathematical programming, signal and image processing.

In the optimization field, the first recurrent neural network model was proposed by Tank and Hopfield in their seminal work as an implementation on analog circuits [Tan86]. Since then many neural network models have been developed to solve different types of optimization problems. Some of them are summarized in Table 1.

Table 1: Different RNN optimization approaches

Recurrent NN	Characteristics	References
Penalty function	Easy implementation, only approximate optimal solutions	[Maa92],[Mla00], [Che09]
Lagrangian	Manage equality and inequality constraints.	[Zha92],[Bou93], [Cos08]
Dual	Overcome penalty-method Lower computational complexity.	[Xia96],[Shu06], [Hu08]
Projection	Without penalty function.	[Xia05],[Che09], [Liu15]
Discrete time	Good for hardware implementation.	[PI13]

Several studies have shown the neurodynamic optimization capability to solve quadratic problems relative to MPC strategy [Que93, Yun08, Yan12, Wan14]. To the best of our knowledge, none of these applications evaluates the applicability of the penalty method as a form of effective optimization for the implementation in an MPC loop.

In this paper, we propose to implement a continuous-time recurrent neural network based gradient method as a QP solver in the platform control of a dynamic driving simulator. A penalty method was proposed to solve the constrained optimization problem in real time. Compared to an active-set method [Goo06], we prove that, by using a saturation function for the decision variables and an appropriate regulation in the energy function, the penalty terms can be avoided and then the optimal solution gives a good performance in a tracking reference task in the MPC.

MPC-based MCA

Problem formulation

The dynamic driving simulator considered in this study is composed of an hexapod mounted on a XY-table. Since it has a dynamic that is predefined by the manufacturer, we consider the double integrator as a perfect model system. The idea with this model is to be able to control for each degree of freedom the longitudinal and rotational accelerations of the platform.

The linear time invariant system is described by the discrete-time state-space model:

$$\begin{aligned} x_m(k+1) &= A_m x_m(k) + B_m u(k) \\ y(k) &= C_m x_m(k) \end{aligned} \tag{1}$$

where $x_m(k) \in \mathbb{R}^s$ is the state vector: position, velocity, $u(k) \in \mathbb{R}^l$ is the input vector: the reference acceleration, $y(k) \in \mathbb{R}^t$ is the output vector: the simulator acceleration, $A_m \in \mathbb{R}^{s\times s}$ and $B_m \in \mathbb{R}^{s\times l}$. The system is subject respectively to the following boundaries conditions:

$$\begin{aligned} x_{min} \leq x(k) \leq x_{max} \\ y_{min} \leq y(k) \leq y_{max} \\ \Delta u_{min} \leq \Delta u(k) \leq \Delta u_{max} \end{aligned} \tag{2}$$

In a general tracking situation, the regulation error and the control action cannot be equal to zero at the same moment, thus we augment the model of the system with an embedded integrator to ensure offset free tracking in the control law [Ros03]. The input is the control increment $\Delta u(k)$ rather than the command $u(k)$ itself:

$$\begin{aligned} x(k+1) &= Ax(t) + B\Delta u(k) \\ y(k) &= Cx(k) \end{aligned} \tag{3}$$

where,

$$\begin{aligned} \overbrace{\begin{bmatrix} \Delta x_m(k+1) \\ y(k+1) \end{bmatrix}}^{x(k+1)} &= \overbrace{\begin{bmatrix} A_m & 0_m^T \\ C_m A_m & I_m \end{bmatrix}}^{A} \overbrace{\begin{bmatrix} \Delta x_m(k) \\ y(k) \end{bmatrix}}^{x(k)} \\ &+ \overbrace{\begin{bmatrix} B_m \\ C_m B_m \end{bmatrix}}^{B} \Delta u(k) \\ y(k) &= \overbrace{\begin{bmatrix} 0_m & I_m \end{bmatrix}}^{C} \begin{bmatrix} \Delta x_m(k) \\ y(k) \end{bmatrix} \end{aligned} \tag{4}$$

The vectors of the predicted output Y and states X and the future control ΔU are created recursively and defined as:

$$\begin{aligned} X &= [\ x(k+1) \quad x(k+2) \quad \cdots \quad x\,(k+N_p)\ ]^T \\ Y &= [\ y(k+1) \quad y(k+2) \quad \cdots \quad y\,(k+N_p)\ ]^T \\ \Delta U &= [\ \Delta u(k) \quad \Delta u(k+1) \quad \cdots \quad \Delta u\,(k+N_u-1)\ ]^T \end{aligned} \tag{5}$$

where N_p denotes the predictive horizon $(1 \leq N_p)$, N_u denotes the control horizon $(0 < N_u \leq N_p)$. In a more compact form:

$$\begin{aligned} Y &= Fx(k) + G\Delta U \\ X &= F_x x(k) + G_x \Delta U \end{aligned} \tag{6}$$

where,

$$G_x = \begin{bmatrix} B & 0 & \dots & 0 \\ AB & B & \dots & 0 \\ A^2B & AB & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ A^{N_p-1}B & A^{N_p-2}B & \dots & A^{N_p-N_u}B \end{bmatrix}$$

$$F_x = \begin{bmatrix} A \\ A^2 \\ \vdots \\ A^{N_p} \end{bmatrix} \tag{7}$$

$$F = CF_x \quad \text{and} \quad G = CG_x \tag{8}$$

In the MPC also known as the receding horizon control, an open-loop control sequence is computed at each sampling time by optimizing an objective function over a finite predictive horizon N_p, within the system constraints (Eq. 2). The first control action is implemented making a closed-loop system. The process is repeated at the next time step, yielding a receding horizon control strategy [Bor07]. The cost function to minimize is of the form:

$$\begin{aligned} J(k) \ = \ & \sum_{j=N_1}^{N_p} \delta(j) \left\| r\,(k+j|k) - y(k+j) \right\|_2^2 \\ & + \sum_{j=1}^{N_u} \lambda(j) \left\| \Delta u(k+j-1) \right\|_2^2 \\ & + \sum_{j=N_1}^{N_p} q(j) \left\| x\,(k+j|k) \right\|_2^2 \end{aligned} \tag{9}$$

where $J(k)$ is a scalar objective, subject to the constraints defined in Eq. 2, $r\,(k+j|k)$ denotes the reference trajectory at sampling instant k, $y(k+j)$ is the predicted output, $\delta(j)$, $\lambda(j)$ and $q(j)$ are weighting parameters for the tracking error, the control rate and the states respectively.

The prediction reference $r(k+j)$ remains constant over the prediction horizon N_p, overcoming the difficulty in predicting the driver's behavior.

In the matrix form, we get

$$J = (Y - R_s)^T Q_\delta (Y - R_s) + \Delta U^T Q_\lambda \Delta U + X^T Q_q X$$

where

$$R_s^T = r(k) \times [1 \quad 1 \quad ... \quad 1]_{1 \times N_p}$$

Replacing the predicted output $Y = Fx\,(k_i) + G\Delta U$ and the predicted states $X = F_x x\,(k_i) + G_x \Delta U$ in J and eliminating all the terms that do not depend on the vector decision variable ΔU, the cost function obtained is

$$\begin{aligned} J \;=\; & \Delta U^T \left(G^T Q_\delta G + Q_\lambda + {G_x}^T Q_q G_x\right) \Delta U \\ & +2 \left(\left(Fx\,(k) - R_s\right)^T Q_\delta G + \left(F_x x\,(k)\right)^T Q_q G_x \right) \Delta U \end{aligned}$$

Finally, the QP problem with the constraints remains:

$$\begin{aligned} &\text{minimize} \quad J = \frac{1}{2} \Delta U^T H \Delta U + f^T \Delta U \\ &\text{subject to} \quad A_c \Delta U \leq b \end{aligned} \tag{10}$$

where,

$$\Delta U \in \mathbb{R}^n, \quad f \in \mathbb{R}^n, \quad b \in \mathbb{R}^m, \quad A_c \in \mathbb{R}^{m \times n}$$

The matrix H is a symmetric positive definite $n \times n$, i.e., all eigenvalues of H are strictly positive and then, the cost function J has a unique global minimum. The matrix equation $A_c \Delta U \leq b$ contains all the linear inequality constraints of Eq. 2. This quadratic problem has to be solved at every sampling time to find the control sequence ΔU and apply the first element of this sequence to the real system as a receding control strategy.

Feasibility of the solution

Many past studies have addressed the feasibility issue at each time step in the optimization problem due to the constraints set on the output and the state constraints. Different solutions exist to address this issue: making the horizon control infinite, setting a terminal state equal to zero, using a discrete linear quadratic regulator terminal condition. The first leads to an optimization problem with infinite constraints that is impossible to solve, the second can considerably reduce the feasible QP solution and might perturb the input trajectory in short horizons. The last one is difficult to apply to a real-time dynamic simulation system [Fan17].

Getting a feasible solution at every sample time requires an extremely high prediction horizon, which is not viable to find the solution in real time. Also previously applied methods are not suitable for MPC-based MCA. To overcome this issue, Fang et al. proposed a different condition [Fan14]. Their proposed approach can verify a solution along the prediction horizon using a braking law once the platform approaches its limits. This law can be compared to an adaptive filter that allows the platform to return to its neutral position while respecting the limits in position, velocity and acceleration. This condition is summarized as:

$$p_{min} \geq p_k + c_v T x + c_a \frac{T^2}{2} \leq p_{max} \tag{11}$$

where the coefficients c_v, c_a and T are tuning parameters that prevent the platform from exceeding its limits. In Table 2 we expose the performance of the dynamic platform we consider here along the degrees of freedom of interest, here the longitudinal and lateral rail-axis.

Table 2: Performance along the longitudinal and lateral rails of Renault ULTIMATE simulator

Rail	Position	Velocity	Acceleration
X	$\pm 2.6m$	$\pm 2m/s$	$\pm 5m/s^2$
Y	$\pm 2.6m$	$\pm 3m/s$	$\pm 5m/s^2$

Recurrent Neural Network model design

The idea of using optimization-based neural networks is to compute a QP optimal solution explicitly and to apply the control action online. The network used in this work is a continuous feedback network. Its dynamics can be depicted by a set of continuous dynamic systems as follows:

$$\dot{x} = w\,(x(t)) \tag{12}$$

For a better understanding, we express the decision vector ΔU in Eq. 10 in terms of the states x in the network represented by Eq. 12.

$$\begin{aligned} &\text{minimize} \quad f(x) = \frac{1}{2} x^T H x + f^T x \\ &\text{subject to} \quad g_i(x) = A_c x - b \leq 0, \quad i = 1, 2...m. \end{aligned} \tag{13}$$

where $f(x)$ is a strongly convex objective function since H is positive definite matrix, $g(x) \in \mathbb{R}^m$ is the inequality constraints. The network represented by Eq. 12 needs a corresponding Lyapunov function $E\,(x(t))$ that defines the stability of the dynamic system trajectory thus, as t increases, the value of $E\,(x(t))$ decreases. It has been proven [Zha13] that the objective function of the optimization problem in Eq. 13 is equivalent to the Lyapunov function of the neural network.

The differential equations in Eq. 12 are solved simultaneously by an associated circuit consisting of highly interconnected processing units called neurons. For some initial state, the network will converge to an equilibrium state that will coincide with the optimal solution to the original problem (Eq. 13).

To solve the QP problem in Eq. 13 and to take into account the system constraints, the next step is to transform the constraint problem into an unconstrained problem to be able to apply a steepest descend method based on the gradient of the function. By using the penalty method, the energy function is adapted to integrate inequality constraints into the

original function.

$$E(x,k) = f(x) + k\sum_{i=1}^{m}\left({g_i}^{+}(x)\right)^2$$
$${g_i}^{+}(x) = \max\{0, g_i(x)\} \tag{14}$$

where $k > 0$ is the constant penalty parameter and ${g_i}^{+}(x)$ is a continuous non-negative penalty function which equals zero at a point if only all constraints are satisfied at that point. The augmented energy function remains a quadratic objective function in which each unsatisfied constraint influences the state x by assigning a penalty multiplied by k. This parameter influences the accuracy of the solution; therefore, it must be large enough to provide a feasible solution. Ideally, when the value of k holds at infinity, the optimal solution of $E(x,k)$ converges to a solution of the original problem with constraints (Eq. 13). In practice this remains impossible to realize since k cannot be set to infinite, therefore, with this method, the solutions are an approximation to the optimization problem (Eq. 13). If k is small, a greater violation of the restrictions will occur since the penalty is not enough to guarantee a feasible solution, on the other hand with a very large value of k the energy function will be computationally ill-conditioned and become a stiff system.

According to the literature [Coc93], it is desirable to create an activation function S_i for the penalty function. That is, every time a constraint is active, the penalty function is different to zero, otherwise it's zero and the solution does not take into account the constraints and it's solve as an unconstrained optimization problem:

$$E(x,k) = f(x) + \frac{k}{2}\sum_{i=1}^{m} S_i[g_i(x)]^2 \tag{15}$$

In order to find the minimal value of the energy function $E(x,k)$, the unsconstrained optimization problem is transformed into an associated system of first-order ordinary differential equations by using the gradient descent method [Coc93]:

$$\frac{dx}{dt} = -\mu\nabla_x E(x,k) \tag{16}$$

The state trajectories $x(t)$ moves in the search direction of the negative of the gradient and will converge to a global minimum (equilibrium point in Eq. 16) for $E(x,k)$, since $E(x,k)$ is a Lyapunov function. The dynamic equation at neuron j can be written

$$\frac{dx_j}{dt} = -\mu\left(\frac{\partial f(x)}{\partial x_j} + k\sum_{i=1}^{m} S_i g_i(x)\frac{\partial g_i(x)}{\partial x_i}\right), \tag{17}$$
$$(j = 1,2,...,N_u), (i = 1,2,...,m), k > 0, \mu > 0$$

$$\mu = diag\left(\mu_1, \mu_2, \ldots, \mu_{N_u}\right), \quad \mu_j = \frac{1}{\tau},$$
$$S_i = \begin{cases} 0, & \text{if } g_i(x) \leq 0 \\ 1, & \text{otherwise} \end{cases}$$

where μ is a learning parameter, τ is the integration time constant of the integrators, m is the number of inequalities constraints. Equations 16 and 17 represent a recurrent neural network as a gradient dynamic system. Replacing the functions by their respective indexes:

$$\frac{dx_j}{dt} = -\mu_j\left(\left(f_j + \sum_{i=1}^{N_u} h_{ji}x_i\right) + k\sum_{i=1}^{m} S_i a_{ij} g_i(x)\right) \tag{18}$$

For convenience, the adaptive gains μ_j are equal. The selection of the appropriate parameter μ is crucial to ensure the stability of the system and the convergence speed to an equilibrium point. Considering the following aspects, the choice of μ is based on a trial and error basis: higher learning rates will decay the loss faster, but if it is too large, the algorithm will become unstable and steepest descent will not converge.

Implementation

To implement the proposed RNN we used Simulink-Matlab. As shown in the previous section, the recurrent neural network is seen as a set of differential equations (16) which represent all the state trajectories x of the network. A simplification of the general topology of the RNN can be seen in Fig. 1. The architecture consists in a continuous-time integrator that can be viewed as a special time-depending neuron; dynamic building blocks seen as multipliers or adders, function generators for the realization of the gradient of the energy and saturated transfer functions.

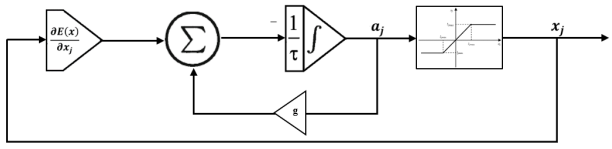

Figure 1: Scheme of the proposed one-layer recurrent neural network

In order to generate code and make a real-time based Simulink application, it is required to use a fixed-step integration algorithm to solve the system in Eq. 16. We simulate the RNN as a optimizer in the control loop as shown in Fig. 5, employing a penalty function method (Eq. 17) for inequality constraints. The method shows a convergence for all the state's trajectories x towards an optimal point since the solution tracking reference is the desired one Fig. 7. Nevertheless, when implementing the available fixed-step time solvers, the method did not work completely since the convergence of Eq. 17 was not ensured for all sample times.

Aiming to find the minimum step that the RNN needs to find a solution, i.e., an equilibrium point for every state x at every sample time, we analyze the step used by a variable-step solver. Fig. 2 shows that the minimum step size to capture the dynamics accurately is very small ($\sim 10^{-8}$) and cannot be applicable to a real time application. As seen in Fig. 2, the differential equation system is stiff, i.e., the system has both slowly and quickly varying continuous dynamics. Therefore it is mandatory to take small time steps in the numerical method to obtain satisfactory results.

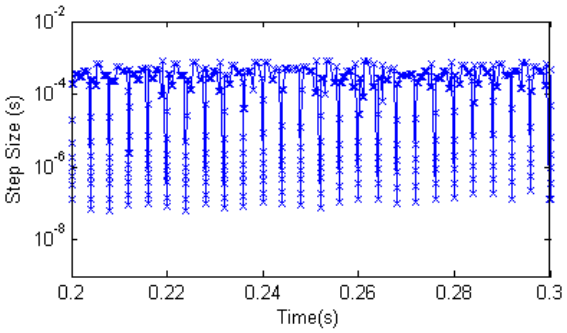

Figure 2: Solver step time for a stiff model

One of the reasons why the differential equations are stiff is that the system is ill-conditioned:

$$cond(H) >> 1 \tag{19}$$

This condition means that the ratio of the largest

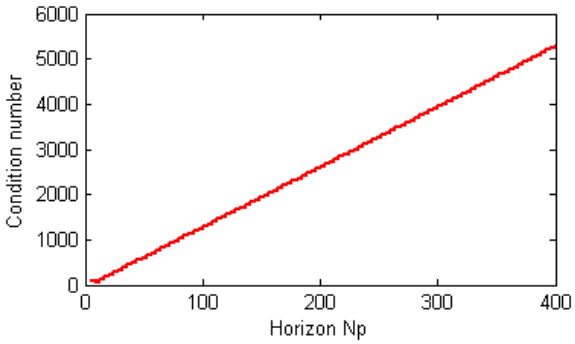

Figure 3: Condition number for the Hessian matrix of the system

eigenvalue to the smallest one of matrix H is very high. In this case, the gradient might not choose the best way to descend to the minimum and the solution to the QP problem may become unreliable. Fig. 3 shows the condition number of the Hessian matrix H as a function of the prediction horizon N_p. However, the energy function does not depend only on the original cost function J (Eq. 13), but also on the penalty function (Eq. 14) that is updated at each sampling step according to the system's constraints $g(x)$ and the states, thus making difficult the preconditioning of the system based solely on the Hessian matrix H of $f(x)$.

Accordingly, to alleviate ill-conditioning and the stiffness of the differential equations, a regulation strategy is implemented. In Fig. 1 we can see a gain g that represents the regularization parameter that helps handling oscillations for each state trajectory x. Additionally, to ensure that all the constraints in Eq.13 are satisfied, and based on the convergence analysis in [Cos08], we convert the constraints for the system to boundary constraints for the control signal ($\Delta U = x$ for the RNN). We choose a saturation linear function depicted in Fig. 4 that ensures the control does not exceed its boundaries:

$$x_j = f(a_j) = \begin{cases} l_{min}, & a_j \leq l_{min} \\ a_j, & a_j \in [l_{min}, l_{max}] \\ l_{max}, & a_j \geq l_{max} \end{cases} \tag{20}$$

$$j = 1, 2, ..., N_u$$

This function will saturate each state trajectory x to

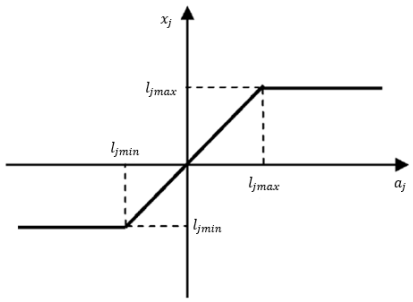

Figure 4: Saturation dynamic function $f(a_j)$

an upper l_{max} or lower l_{min} limit based on the constraints and the actual dynamic states of the platform at each simple time. The RNN will find at a fixed-step time a control sequence $\Delta U = x$ for which the gradient of the cost function is zero. When the dynamic platform approaches the limits that are governed by the restrictions on the system, the saturation function will prevent the gradient from finding an unstable solution and will limit the value to the maximum or the minimum gradient as shown in Eq. 21.

$$\begin{aligned} x_j = l_{max}, \quad & \frac{\partial E(x)}{\partial x_j} \leq 0 \\ l_{min} < x_j < l_{max} \quad & \frac{\partial E(x)}{\partial x_j} = 0 \\ x_j = l_{min}, \quad & \frac{\partial E(x)}{\partial x_j} \geq 0. \end{aligned} \tag{21}$$

The proposed dynamic system describing this behavior is detailed in the following state equations:

$$\frac{dx}{dt} = -\mu \left(\nabla_x E(x) + g.a \right) \tag{22}$$

Since the Hessian matrix H of the function $f(x)$ is positive-definite, each equilibrium point of the system (Eq. 22) corresponds to an optimal global solution of (Eq. 13).

It is important to mention that the recurrent neural network used in this work (Fig. 1) has N_u neurons-based continuous time integrators that considerably improve the computational cost and allow us easy implementation. Fig. 5 shows the block diagram of the proposed control scheme.

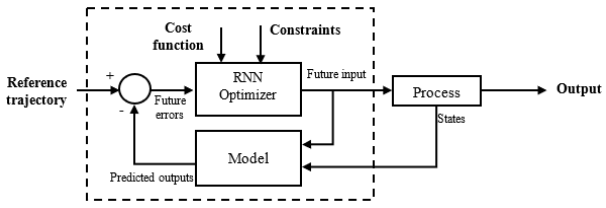

Figure 5: Block diagram of our proposed control scheme

Control Scheme

Based on the recurrent neural network design described above and making reference to the MPC-based MCA control strategy (Fig. 5), the control scheme can be summarized as follows:

1. Define the prediction horizon N_p, the control horizon N_u, the sample time, the weighting parameters $\delta(j)$, $\lambda(j)$ and $q(j)$ in Eq. 9.
2. Transform Eq. 9 in a QP problem (Eq. 13) and update the matrices H, f, A, b with the actual states and constraints.
3. Solve the quadratic programming problem using the recurrent neural network approach by calculating the optimal control actions ΔU.
4. Calculate the optimal control input $u(k) = \Delta u + u(k-1)$.
5. Apply only the first element in $u(k)$ sequence and then apply it as an input to the system.

Simulation and results

Using the SCANeR Studio simulation software†, a driving simulation scenario was generated to take the vehicle information as the acceleration signal. The idea is to analyze the capacity of the simulator response to respect the limits and tracking the signal using the MPC-based MCA method with an optimization based on neural networks.

The scenario consists in a simple maneuver remaining at an initial position for four seconds then following a car with a speed of $30\ m/s$ in a straight line. The scenario lasts only 18 seconds which is enough to evaluate the behavior of the system as can be seen in the reference signal in Fig. 7.

A fixed-time implicit solver was implemented, since it provides greater stability for oscillatory behaviors, avoiding unnecessary computational cost. The global sampling time of the system was set to 1 ms, and a minimum of two iterations are required for the network to converge to an optimal solution. The platform system is sampled with a 10 ms step that is the minimum required for a real-time application (control frequency at 100 Hz).

For the QP problem (Eq. 10), the tuning parameters were the followings: $d = 1$, $\lambda = 0.1$, $q(pos) = 1$, $q(vel) = 0.1$, $q(acc) = 0$, $N_u = 3$, $N_p = 100$. Optimization is implemented on each independent degree of freedom of the platform system, but here we only show the simulation of the most relevant axis: the longitudinal axis.

First, we conducted a comparison to validate the accuracy of the optimal value at each time step. A quadprog-Matlab function is used as the comparison solver. This function can apply tree different algorithms: trust region reflective, interior point convex or active set. We decided to implement an active set method to find the QP optimal solution at each sample time. Using the previous information, we found that the error between the solution of that solver and the proposed approach based on RNN is almost null. In Fig. 6, we can see that the platform acceleration given by the control input of both curves overlap, proving the applicability of the RNN to solve the MPC-based MCA optimization problem.

† http://www.oktal.fr/en/automotive/range-of-simulators/software

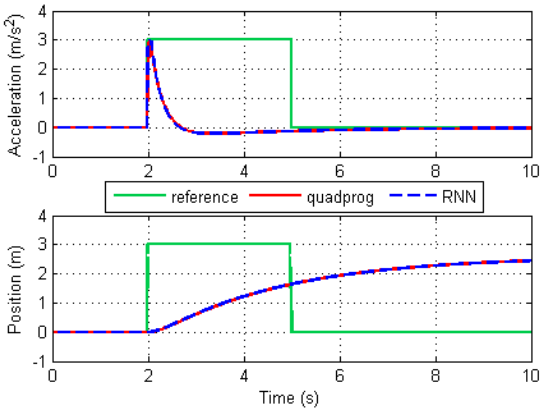

Figure 6: Comparison between quadprog and the proposed RNN

Secondly, we analyzed the average time the quadprog takes to solve the optimization problem. This analysis was based using the model profiler application available in Matlab-Simulink. We found that for our problem with 3 states the quadprog function takes at least 3 ms to find a solution. In the simulink profile report can be seen the time taken by each block in solving the specific operation. For the optimization subsystem (Fig. 1) the total time for one sample time is $0.235ms$. As stated above, a minimum of 3 iterations is required to find a optimal solution, then the time required to solve the Qp problem is $\sim 0.7ms$. This comparison shows the potential of the proposal approach to solve problems with a greater number of states and solve the MPC-MCA obtimization problem in real time respecting the plateform constraints.

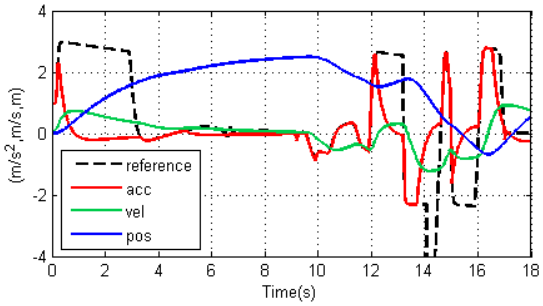

Figure 7: Longitudinal acceleration rendering using rails only

Fig. 7 shows the longitudinal (X-axis) signals of the motion system obtained by the resulting strategy and the actual platform acceleration during the scenario. It is observed that the platform tries to follow the reference of the acceleration until it approaches the limits. Then, the restrictions are applied so that the optimization can find a solution at each sampling time. The way in which the acceleration behaves when it approaches the limits depends on the parameters mentioned in the feasibility section.

Comparing with previous recurrent neural networks based on penalty functions (Tab. 1) for solving a quadratic optimization problem with bounds constraints, our method shows to have lower computa-

tional complexity since the penalty function can be replaced by using a saturated activation function for the bounds constraints and an appropriate regularization parameter. Consequently, the number of neurons and logical operations used to solve the QP problem can be reduced.

Conclusions and future work

The resulting set of differential equations requires less components with respect to existing neural networks, for the same class of optimization problems, and could converge to the optimal solution in a finite time. The simulation results confirm the effectiveness of our proposed method and better performance can be obtained with an appropriate learning rate tuning. Our method provides a good tracking performance with fast responses without exceeding the system limits in position, velocity and acceleration.

The neural networks parallel capability, gives a great opportunity for the implementation of this technique for a more complex system. Using a multi-core implementation, we can eliminate the adaptive filter condition of the constraints with a prediction horizon long enough to ensure optimization feasibility, avoiding the tuning task and making the algorithm much more robust.

As future works, we aim at tuning the learning rate so that the system can find a solution as quickly as possible within the required time and with an increased number of states. It is also intended to integrate the different axes of the platform, e.g., combine optimization between the longitudinal axis and the pitch angle. Future development will include a validation test of the algorithm on the real setup.

References

A. Beghi, M. Bruschetta and F. Maran, **A real time implementation of MPC based Motion Cueing strategy for driving simulators**, in Decision and Control (CDC), 2012 IEEE 51st Annual Conference on, 6340–6345, IEEE, 2012.

C. Bordons and E. Camacho, **Model predictive control**, Springer Verlag London Limited, 2007.

A. Bouzerdoum and T. R. Pattison, **Neural network for quadratic optimization with bound constraints**, *IEEE transactions on neural networks*, vol. 4(2): 293–304, 1993.

L. Cheng, Z.-G. Hou, N. Homma, M. Tan and M. M. Gupta, **Solving convex optimization problems using recurrent neural networks in finite time**, 538–543, IEEE, 2009, ISBN 978-1-4244-3548-7.

A. Cochocki and R. Unbehauen, **Neural networks for optimization and signal processing**, John Wiley & Sons, Inc., 1993.

G. Costantini, R. Perfetti and M. Todisco, **Quasi-Lagrangian Neural Network for Convex Quadratic Optimization**, *IEEE Transactions on Neural Networks*, vol. 19(10): 1804–1809, 2008.

M. Dagdelen, G. Reymong and N. MaÃ¯zi, **MPC Based motion cueing algorithm: developpement and application to the ULTIMATE driving simulator**, DSC 2004 Europe, 2004.

Fang, zhou and Kemeny, Andras, **Motion cueing algorithms for a real-time automobile driving simulator**, 2012.

Z. Fang and A. Kemeny, **Review and prospects of Renault's MPCbased motion cueing algorithm for driving simulator**, Paris, 2014.

Z. Fang, M. Tsushima, E. Kitahara, N. Machida, D. Wautier and A. Kemeny, **Motion cueing algorithm for high performance driving simulator using yaw table**, *IFAC-PapersOnLine*, vol. 50(1): 15965–15970, 2017.

N. J. Garrett and M. C. Best, **Driving simulator motion cueing algorithms- a survey of the state of the art**, 2010.

G. Goodwin, M. M. Seron and J. A. De DonÃ¡, **Constrained control and estimation: an optimisation approach**, Springer Science & Business Media, 2006.

X. Hu and J. Wang, **An Improved Dual Neural Network for Solving a Class of Quadratic Programming Problems and Its -Winners-Take-All Application**, *IEEE Transactions on Neural Networks*, vol. 19(12): 2022–2031, 2008.

Q. Liu and J. Wang, **A Projection Neural Network for Constrained Quadratic Minimax Optimization**, *IEEE Transactions on Neural Networks and Learning Systems*, vol. 26(11): 2891–2900, 2015.

C.-Y. Maa and M. A. Shanblatt, **Linear and quadratic programming neural network analysis**, *IEEE transactions on neural networks*, vol. 3(4): 580–594, 1992.

V. M. Mladenov and N. Maratos, **Neural networks for solving constrained optimization problems**, *Proc. of CSCC'00, Athens, Greece,(N. Mastorakis*, 2000.

L. Nehaoua, H. Arioui, S. Espie and H. Mohellebi, **Motion cueing algorithms for small driving simulator**, 3189–3194, IEEE, 2006, ISBN 978-0-7803-9505-3.

Y. Pan and J. Wang, **Two neural network approaches to model predictive control**, 1685–1690, IEEE, 2008, ISBN 978-1-4244-2078-0.

M. J. Perez-Ilzarbe, **New Discrete-Time Recurrent Neural Network Proposal for Quadratic Optimization With General Linear Constraints**, *IEEE Transactions on Neural Networks and Learning Systems*, vol. 24(2): 322–328, 2013.

J. Quero, E. Camacho and L. Franquelo, **Neural network for constrained predictive control**, *IEEE Transactions on Circuits and Systems I: Fundamental Theory and Applications*, vol. 40(9): 621–626, 1993.

J. A. Rossiter, **Model-based predictive control: a practical approach**, Control series, CRC Press, 2003, ISBN 978-0-8493-1291-5.

Shubao Liu and Jun Wang, **A Simplified Dual Neural Network for Quadratic Programming With Its KWTA Application**, *IEEE Transactions on Neural Networks*, vol. 17(6): 1500–1510, 2006.

D. Tank and J. Hopfield, **Simple'neural'optimization networks: An A/D converter, signal decision circuit, and a linear programming circuit**, *IEEE transactions on Circuits and Systems*, vol. 33(5): 533–541, 1986.

J. Venrooij, D. Cleij, M. Katliar, P. Pretto, H. Bulthoff, D. Steffen, F. Hoffmeyer and H. Schoner, **Comparison between filter-and optimization-based motion cueing in the Daimler Driving Simulator**, 2016.

X. Wang, Z. Yan and J. Wang, **Model predictive control of multi-robot formation based on the simplified dual neural network**, 3161–3166, IEEE, 2014, ISBN 978-1-4799-1484-5 978-1-4799-6627-1.

Y. Xia, **A new neural network for solving linear and quadratic programming problems**, *IEEE transactions on neural networks*, vol. 7(6): 1544–1548, 1996.

Y. Xia and G. Feng, **An improved neural network for convex quadratic optimization with application to real-time beamforming**, *Neurocomputing*, vol. 64: 359–374, 2005.

Z. Yan and J. Wang, **Model Predictive Control of Nonlinear Systems With Unmodeled Dynamics Based on Feedforward and Recurrent Neural Networks**, *IEEE Transactions on Industrial Informatics*, vol. 8(4): 746–756, 2012.

Yunpeng Pan and Jun Wang, **Nonlinear model predictive control using a recurrent neural network**, 2296–2301, IEEE, 2008, ISBN 978-1-4244-1820-6.

S. Zhang and A. Constantinides, **Lagrange programming neural networks**, *IEEE Transactions on Circuits and Systems II: Analog and Digital Signal Processing*, vol. 39(7): 441–452, 1992.

X.-S. Zhang, **Neural networks in optimization**, vol. 46, Springer Science & Business Media, 2013.